(BASE + OFFSET) == (BASE | OFFSET)

Fig. 1

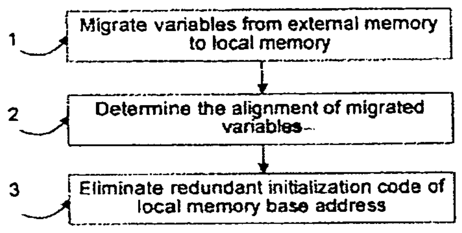

1
Migrate variables from external memory to local memory
2
Determine the alignment of migrated variables
3
Eliminate redundant initialization code of local memory base address

Fig. 2

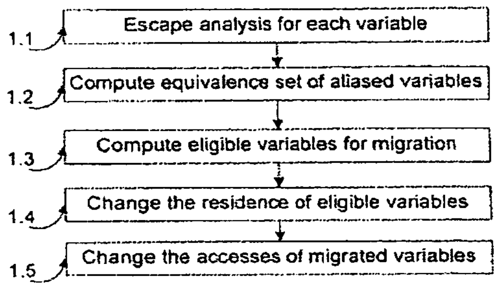
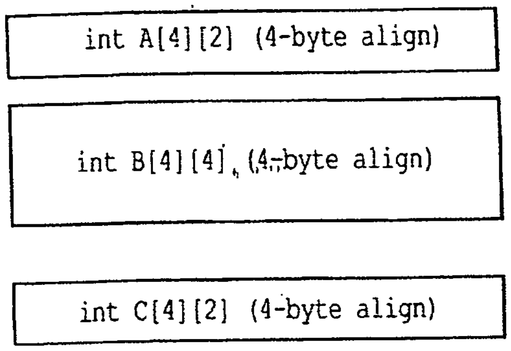

1.1
Escape analysis for each variable
1.2
Compute equivalence set of aliased variables
1.3
Compute eligible variables for migration
1.4
Change the residence of eligible variables
1.5
Change the accesses of migrated variables

Fig. 3


int A[4][2] (4-byte align)
int B[4][4] (4-byte align)
int C[4][2] (4-byte align)

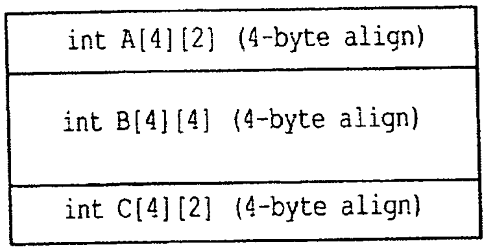
Original Data

FIG. 4A

```
1 Access Address A[i][0]
2 Access Address A[i][1]
3 Access Address B[i][0]
4 Access Address B[i][1]
5 Access Address B[i][2]
6 Access Address B[i][3]
7 Access Address A[i][0]
8 Access Address A[i][1]
```

Pseudo code sequence of accessing A, B, C

FIG. 4B


int A[4][2] (4-byte align)
int B[4][4] (4-byte align)
int C[4][2] (4-byte align)

Data in local memory

FIG. 5A

```
Set the base address to A[i][0]
Access Address A[i][0] (A[i][0]+0)
Set the base address to A[i][1]
Access Address A[i][1] (A[i][1]+0)
Set the base address to B[i][0]
Access Address B[i][0] (B[i][0]+0)
Set the base address to B[i][1]
Access Address B[i][1] (B[i][1]+0)
Set the base address to B[i][2]
Access Address B[i][2] (B[i][2]+0)
Set the base address to B[i][3]
Access Address B[i][3] (B[i][3]+0)
Set the base address to C[i][0]
Access Address C[i][0] (C[i][0]+0)
Set the base address to C[i][1]
Access Address C[i][1] (C[i][1]+0)
```

Pseudo code sequence of accessing A, B, C with initialization code of local memory based address inserted

FIG. 5B

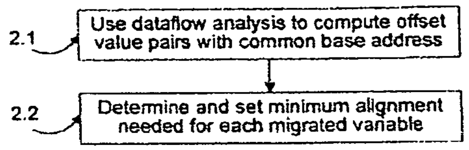
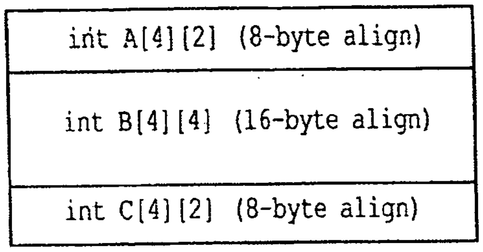

2.1
Use dataflow analysis to compute offset value pairs with common base address
2.2
Determine and set minimum alignment needed for each migrated variable

Fig. 6

```
Compute the GEN, KILL, IN, and OUT for each flow node, fill in the hash table (base address, set of offset value pair)
for (each base address in the hash table)
  VAR is the variable accessed by the base address
  for (each offset value pair for this base address)
    int CURR_VAR_ALIGN = current alignment of VAR
    int CURR_BASE_ALIGN = current alignment of the base address
    if (CURR_BASE_ALIGN does not satisfy the condition in Figure 1 for one of the offset value in the pair)
      int NEEDED_BASE_ALIGN = the minimum base address alignment needed to satisfy the condition in Figure 1 for all offset values in the pair
      int new_align = CURR_VAR_ALIGN * NEEDED_BASE_ALIGN / CURR_BASE_ALIGN
      if (new_align <= MAX_ALIGN(VAR))
        set the alignment of VAR to new_align
        if (the alignment change does not make the base address satisfy the condition in Figure 1 for all offset values in the pair)
          restore VAR's alignment to CURR_VAR_ALIGN
        end if
      end if
    end if
```

Fig. 7


int A[4][2] (8-byte align)
int B[4][4] (16-byte align)
int C[4][2] (8-byte align)

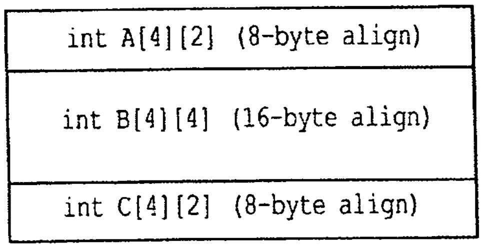
Data with adjusted alignment

FIG. 8A

```
Set the base address to A[i][0]
Access Address A[i][0] (A[i][0]+0)
Set the base address to A[i][0]
Access Address A[i][1] (A[i][0]+4)
Set the base address to B[i][0]
Access Address B[i][0] (B[i][0]+0)
Set the base address to B[i][0]
Access Address B[i][1] (B[i][0]+4)
Set the base address to B[i][0]
Access Address B[i][2] (B[i][0]+8)
Set the base address to B[i][0]
Access Address B[i][3] (B[i][0]+12)
Set the base address to C[i][0]
Access Address C[i][0] (C[i][0]+0)
Set the base address to C[i][0]
Access Address C[i][1] (C[i][0]+4)
```

Pseudo code sequence of accessing A, B, C after insert code to initialize the local memory base address

FIG. 8B


int A[4][2] (8-byte align)
int B[4][4] (16-byte align)
int C[4][2] (8-byte align)

Data with adjusted alignment

FIG. 9A

```
Set the base address to A[i][0]
Access Address A[i][0] (A[i][0]+0)
Access Address A[i][1] (A[i][0]+4)
Set the base address to B[i][0]
Access Address B[i][0] (B[i][0]+0)
Access Address B[i][1] (B[i][0]+4)
Access Address B[i][2] (B[i][0]+8)
Access Address B[i][3] (B[i][0]+12)
Set the base address to C[i][0]
Access Address C[i][0] (C[i][0]+0)
Access Address C[i][1] (C[i][0]+4)
```

Pseudo code sequence of accessing A, B, C after insert code to initialize the local memory base address

FIG. 9B

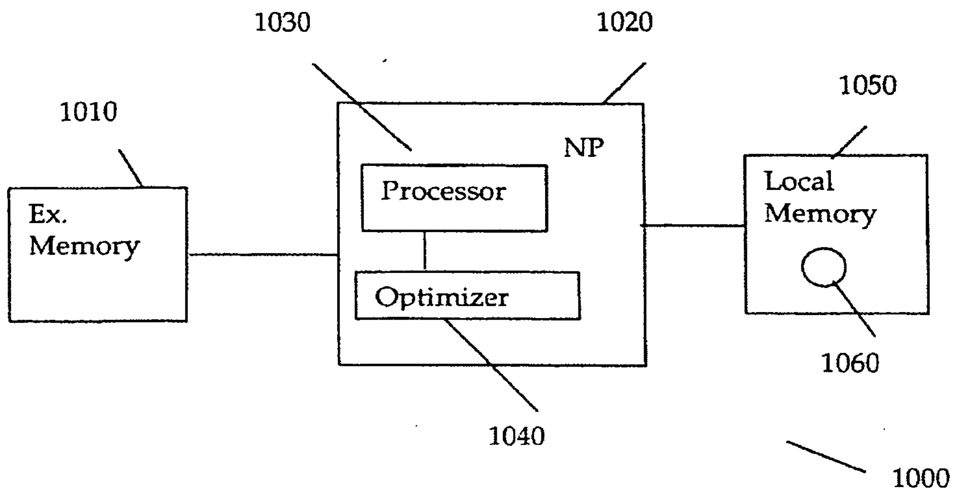

1030
1020
1050
1010
NP
Processor
Ex. Memory
Local Memory
Optimizer
1060
1040
1000

Fig. 10

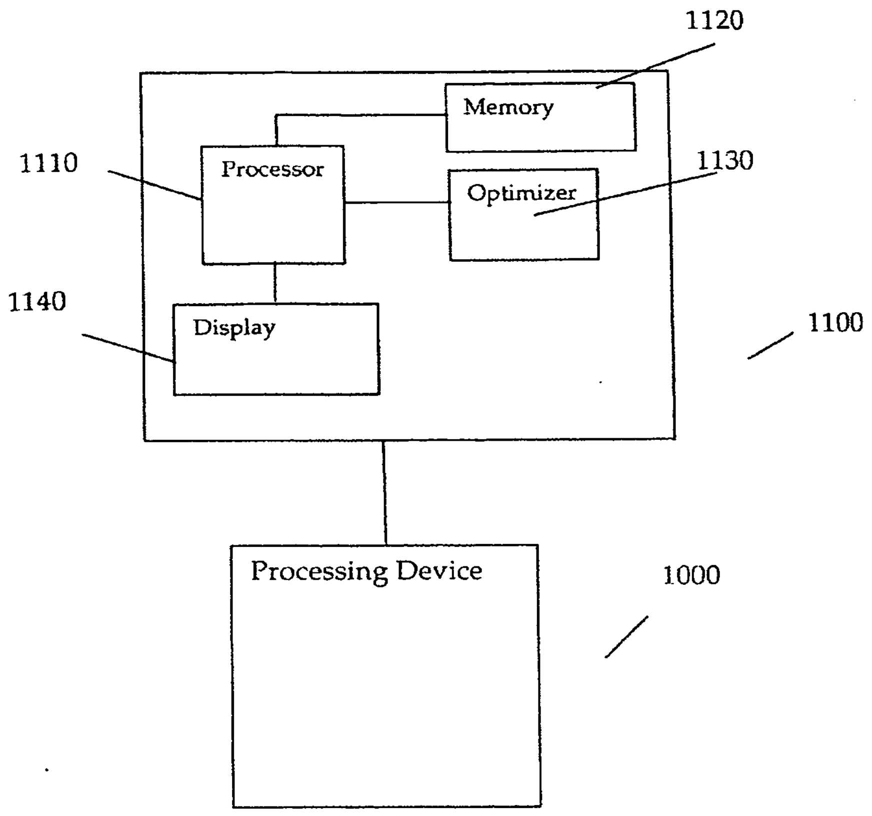

1120
Memory
1110
Processor
1130
Optimizer
1140
Display
1100
Processing Device
1000

Fig. 11

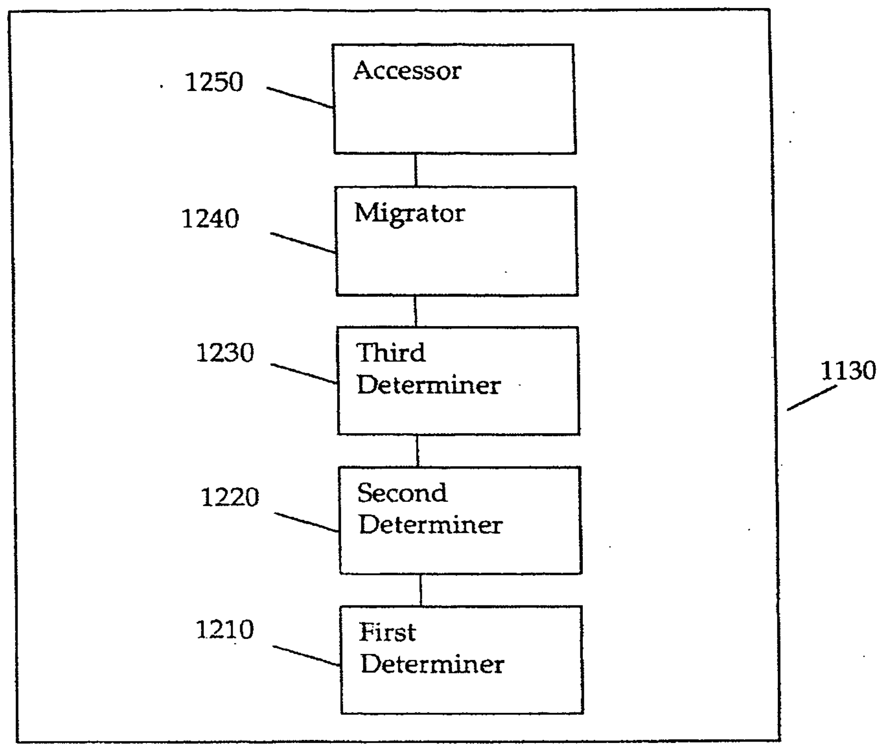

1250
Accessor
1240
Migrator
1230
Third Determiner
1130
1220
Second Determiner
1210
First Determiner

Fig. 12